TGS-Prüfungsfragen
(bis SoSe2015)

1.

Thema 1 [Einführung]

1. Sie würfeln zweimal nacheinander **mit jeweils zwei** Würfeln und notieren die Summe der Augen (sum of dots) als Zahl x. Beim ersten Wurf ist $x_1 = 2$, beim zweiten Wurf ist $x_2 = 11$.
   Hinweis: Überlegen Sie jeweils, wie hoch die Auftrittswahrscheinlichkeiten (für $x_1 = 2$ bzw. $x_2 = 11$) sind.

   Welche der folgenden Aussagen ist richtig? **Jeweils kurze Begründung!**

   a. Der Informationsgehalt von x_2 ist …

   ☐ doppelt so groß wie der von x_1 ☐ gleich groß wie der von x_1

   ☐ 1 bit weniger als der von x_1 ☐ 1 bit mehr als der von x_1

   Begründung:

b. Der Informationsgehalt von x_2 beträgt …

☐ 2 bit ☐ zwischen 2 und 3 bit ☐ zwischen 3 und 4 bit ☐ mehr als 4 bit

Begründung:

2. **Informationsgehalt:** Sie lesen einen Buchstaben (Annahme: **nur** Kleinbuchstaben): Wie hoch ist der Informationsgehalt der Buchstaben „e“ und „f“ in Bit (Shannon) wenn …

   a. … alle Buchstaben gleich-wahrscheinlich sind (`auswählen`).

   ☐ ≈ 4,0 Bit ☐ ≈ 3,7 Bit ☐ ≈ 4,7 Bit ☐ ≈ 2,6 Bit

   kurze Begründung:

b. ... die Auftrittswahrscheinlichkeit P(„e“) = 12,5% und P(„f“) = 1/32 betragen (**berechnen**).

3. Berechnen Sie den Informationsgehalt des Wortes „fee“ in Bit auf Basis der in 1b) angegebenen Wahrscheinlichkeiten (unter der **Annahme**, dass die Buchstaben unabhängig voneinander auftreten).

Berechnen Sie B+!B sowie B+!(B-1) und interpretieren Sie die Ergebnisse.
B ist eine 8-Bit Binärzahl, die dem Wert 15 im Dezimalsystem entspricht.
(!(B-1) = Einerkomplement von B)

Zweierkomplement:

$$\begin{array}{rr} \boldsymbol{B =} & \mathbf{0000\ 1111} \\ \hline \boldsymbol{NOT =} & \mathbf{1111\ 0000} \\ \mathbf{+1} & \mathbf{1} \\ \hline \boldsymbol{!B =} & \mathbf{1111\ 0001} \\ \boldsymbol{B =} & \mathbf{0000\ 1111} \\ \hline & \mathbf{1\ 0000\ 0000} \end{array}$$

Übertrag wird gestrichen → Ergebnis ist 0, da bei dem Zweierkomplement zu der negierten Zahl 1 addiert wird

Einerkomplement:

$$\begin{array}{rr} \boldsymbol{B =} & \mathbf{0000\ 1111} \\ \hline \boldsymbol{NOT =} & \mathbf{1111\ 0000} \\ \hline & \mathbf{1111\ 1111} \end{array}$$

Wenn man das Kompliment einer Zahl im Einerkomplement zu der Zahl addiert bekommt man lauter 1, was im Einerkomplement der -0 entspricht

Begründen Sie, wie viele Bit (mindestens) benötigt werden um 74 unterschiedliche Zustände darstellen zu können.

- wird über den ld(Zustände) berechnet:
- $n = ld(74) = 6.21$
- da: 6 < n < 7 → es werden mindestens 7 bit gebraucht.
- Andere Methode: 2^6 (64) < 74 < 2^7 (128)

Erklären Sie den Unterschied zwischen Kilobyte und Kibibyte

- Kilobyte = 10^3 Byte (1000)
- Kibibyte = 2^10 Byte (1024)

- Allgemein:
  - Kilo, Mega, Giga, usw.: Basis 10
  - Kibi, Mebi, Gibi, usw.: Basis 2

2. Komplemente im Oktalsystem. Gegeben: $x = 1230_8$ und $y = 4567_8$. Berechnen Sie $y - x$ mit Hilfe des (vollständigen) Achterkomplements.

3. Warum wird der Exponent einer Fließkommazahl (IEEE-754) mittels Exzessdarstellung gespeichert? Gibt es Ihrer Meinung nach sinnvolle Alternativen zur Exzessdarstellung, um den Exponent zu speichern?

1.(I, II) **Was ist ein Floating Gate?**
Wozu wird es verwendet?

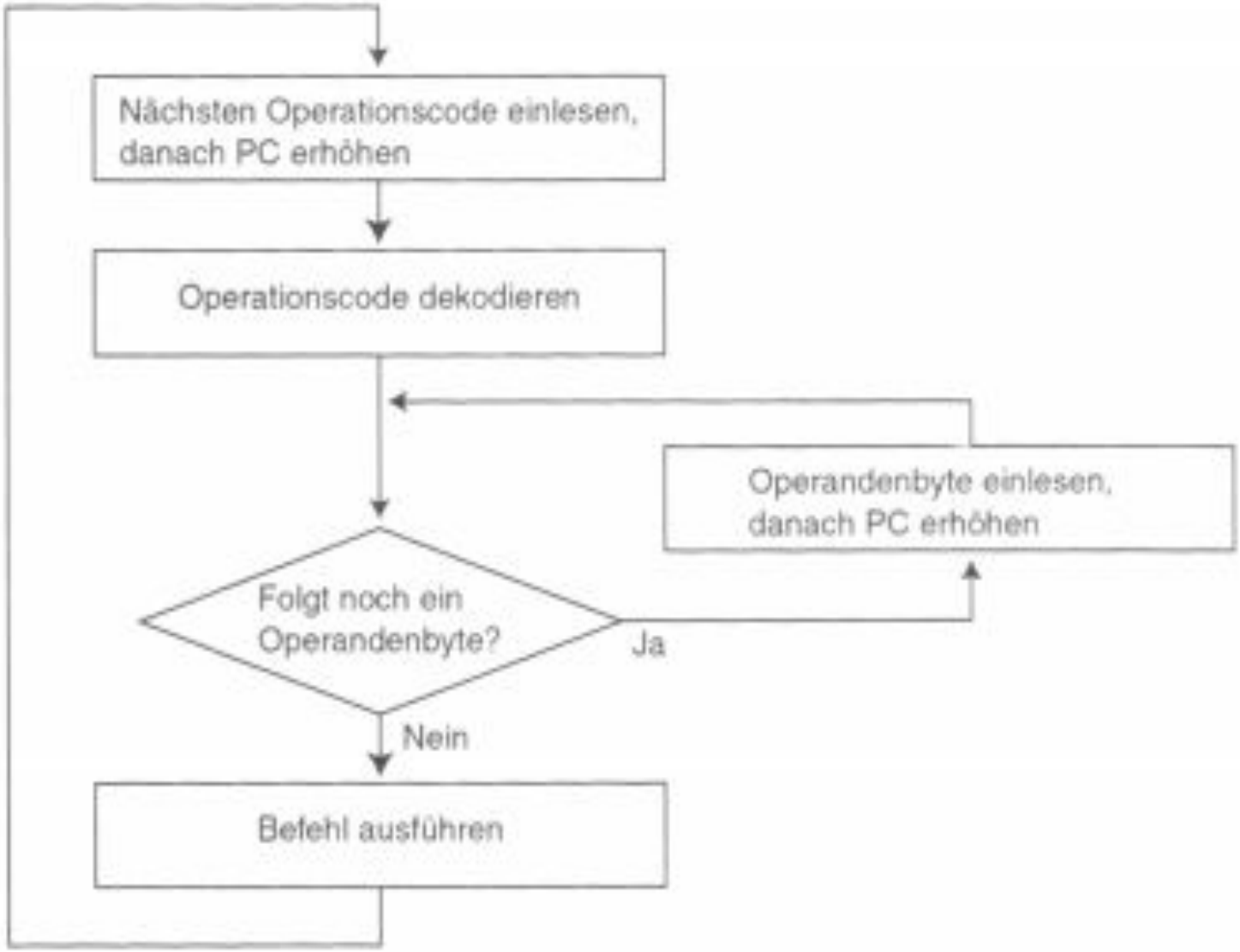
- wird bei PROM, EPROM, EEPROM, sowie Flash Speicher verwendet
  - EPROM kann durch UV – Licht neu programmiert werden
  - EEPROM kann elektrisch entladen und damit neu programmiert werden

2. (I, II)**Erklären sie den Fetch - Decode - Execute Zyklus**.

Was passiert in jedem dieser einzelnen Schritte?

- Maschinencode wird in Programmspeicher geladen
- **Fetch:** Nächster auszuführender OpCode wird aus Program (auch Instruction) Counter gelesen Spezialregister das nächsten Befehl enthält
- **Decode:** OpCode wird bitweise mit bekannten Mustern (**Befehlsatz**) verglichen, um Bedeutung herauszufinden
- Falls OpCode mit Operand, wird Programm Counter inkrementiert, um Operanden auf nachfolgenden Speicherplatz zu lesen (bis alle Operanden gelesen sind)
- **Execute:** OpCode wird ausgeführt; Programm Counter (PC) wird inkrementiert und zeigt damit auf den nächsten Befehl

Erklären sie den Fetch – Decode – Execute Zyklus (FDE). Was passiert in jedem dieser Schritte?



3.(I) Amdahl‘s Law

- Angenommen die Fraction Enhanced beträgt 1/4 der Gesamtausführzeit und der Speedup Enhanced (für dieses 1/4) geht gegen unendlich.
  Wie hoch ist der größtmögliche Gesamtspeedup?

$$S = \frac{1}{(1 - F_E) + \frac{F_E}{S_E}}$$

- Fraction Enhanced (F_E)
Anteil der Gesamt ET der verbessert ist. $F_E \leq 1$
- Speedup Enhanced (F_E)
Faktor um den F_E verbessert ist. $F_E > 1$

4.(II) Amdahl‘s Law

Variante2:

- Fraction Enhanced 2/3. Wie groß muss der Speedup Enhanced (für diese 2/3) mindestens sein, damit sich der Gesamtspeedup des Systems verdoppelt?

5.(I) Erklären Sie den Zusammenhang zwischen Big – und Little Endian und sagen Sie welche Variante im Alltag vorkommt (mit Bsp). (msb/lsb;MSB/LSB;Big/Little Endian)

Erklären Sie msb/lsb, MSB/LSB, Big/Little Endian

msb = most significant bit = Höchstwertigster bit

lsb = least significant bit = Niedrigwertigster bit

MSB = most significant Byte = Höchstwertigster Byte

LSB = least significant Byte = Niedrigwertigster Byte

Big und Little Endian entscheiden wo der höchstwertige Teil steht

- Big Endian: höchstwertige Teil steht **links**
- Little Endian: höchstwertige Teil steht **rechts**

Angenommen 1000 ist Big Endian, wäre 1 MSB und 0 LSB

Im Alltag kommt Big Endian z.B. bei Zahlen vor:

- 1245:
  - 1 □ höchstwertiger Teil (Tausenderstelle)
  - 5 □ niederwertigster Teil (Einerstelle)

6.(II) Konvertieren Sie 1101 1011 01112 in das Hexadezimalsystem. Wie groß ist diese Zahl im Dezimalsystem ungefähr

a) <1000
b) Zwischen 1001 und 2000
c) Zwischen 2001 und 3000
d) > 3000?

- Hex. Lösung: CA7
- Dezimal soll größer >3000 sein

Was hat das Binäre-, Oktale- und Hexadezimale Zahlensystem gemeinsam? Wie geht die Formel dazu? Zu welchem Zahlensystem gehören sie?

- alle drei Systeme haben als Basis ein Potenz von 2
  - Binär: B=2
  - Oktal: B=2^3=8
  - Hexadezimal: B=2^4=16
  - Formel für **Stellenwertsysteme**:

$$x = \sum_{i=0}^{n-1} b_i \cdot B^i = b_0 \cdot B^0 + b_1 \cdot B^1 + b_2 \cdot B^2 + \cdots + b_{n-1} \cdot B^{n-1}$$

  - alle drei Systeme gehören zu den **Stellenwertsystemen**

Wie wird im Dualsystem multipliziert/dividiert? Welche logische Operation steht dem zu?

- Multiplikation:
  - Zahl wird um S Stellen nach links verschoben
  - x * B^S
  - Vorsicht vor Überläufen (Overflow)

- Division:
  - Zahl wird um S Stellen nach rechts verschoben
  - x * B^(-S)
  - Vorsicht vor Unterläufen

7.(I) **Wann werden Komplemente in der Zahlensdarstellung am Computer verwendet?**

Nenne Sie mindestens zwei Komplemente

8.(I) Single Precision

Abbildung 1 zeigt alle 32 Bit einer Fließkommazahl bei Single Precision (IEEE 754 Standard).
Erklären Sie die zwei Felder nach dem Vorzeichen (VZ) die durch dicke Balken getrennt sind. Welche Information der Fließkommazahl wird in diesen beiden Feldern gespeichert? **(2 P)**

Abbildung 1: Single Precision

32 Bits Einfache Genauigkeit

- Exponent (8bit) , für Bereich entscheidend

und Mantisse(23bit) , für Genauigkeit entscheidend

9.(II) **Nennen Sie zwei Stellenwertsysteme, die in der Informatik oft genutzt werden. Wodurch wird bei Stellenwertsystemen der Wert einer Zahl bestimmt?**

Dezimalsystem, binärsystem, usw.

Wert einer Zahl durch Form und Position der Zeichen (Symbole) bestimmt (s.8)

10.(I) **Kann der rechnerisch wirkende Exponent einer Fließkommazahl (IEEE 754 Standard) kleiner als 0 sein?**
Wenn ja, wie kann dieser negative Wert gespeichert werden?
Geben sie ein Beispiel an, bei dem der rechnerisch wirkende Exponent kleiner als 0 ist!

- Ja, einfach eine Negative zahl ins IEEE 754 format umrechnen. (es geht um den rechnerischen exponent also zahl * 10^-X)

Argumentieren Sie, warum folgende Aussagen korrekt bzw. falsch sind (ohne Begründung keine Punkte).

- **Mit double Precision kann man größere Zahlen speichern bzw. verarbeiten als mit Single Precision.**
- **Korrekt –** Man hat 11 bit statt 8 bit für den Exponenten, man kann das Komma um ca. 900 Stellen weiter verschieben als mit Single Precision!
- **Mit Double Precision treten im Gegensatz zu Single Precision keine Rundungsfehler mehr auf.**
- **Falsch –** Es können noch immer Rundungsfehler auftreten, jedoch sind diese bei Double Precision deutlich geringer (Single ca. 7 Dezimalstellen; Double ca. 15 Dezimalstellen)
- **Der rechnerisch wirkende Exponent einer Fließkommazahl (IEEE 754) kann nicht kleiner 0 sein.**
- **Falsch –** Er kann kleiner 0 sein, ansonsten könnte man keine Zahlen wie 0,0000321 darstellen, da diese in der Gleitkommadarstellung einen rechnerisch wirkenden Exponenten von -5 hätte.

11.(I) **Erklären Sie, warum bei der Verarbeitung von reellen Zahlen am Computer Rundungsfehler auftreten können.**
Welcher Teil der Fließkommazahl ist hier betroffen? Welche Möglichkeit gibt es, diese Rundungsfehler zu minimieren? Entsteht dadurch (Minimierung der Rundungsfehler) irgendein Nachteil?

- Wegen Begrenzte Anzahl von Bits können nur bestimmte Werte dargestellt werden.
  infos in mikroprozessortechnik Buch Seite17.

12.(II) **Wie viele unterschiedliche Zustände können mit n Bit dargestellt werden?**

- 2 auf n Zustände (s.18)

13.(II) **Erklären Sie die Bedetung von Exponent und Mantisse bei Fließkommazahl-Darstellung**
(IEEE 754 Standard)

- Siehe Frage 8

Weitere Prüfungsfragen
(Thema2 – Boolesche Algebra)

15.(I) **Welche der folgenden 3 Booleschen Ausdrücke sind identisch?**

Hinweis: Versuchen Sie alle Terme auf eine der Normalformen zu bringen!

- DNF: (not a and b) or not c
- Alle drei

Warum werden Normalformen verwendet?

- um Ausdrücke zu vereinfachen und leserlicher zu machen, da viele verschiedene Formen der Darstellung von logischen Aussagen möglich ist

- z.B.: bei Computern: um Gatterschaltungen zu vereinfachen

Welche beiden wichtigen Normalformen (für Boole´sche Funktionen) gibt es (vollständige Bezeichnung)?

- **KNF = Konjunktive Normalform**
  - Volldisjunktionen werden konjunktiv verknüpft „(x oder y) und (x oder z)“
  - Wahrheitstabelle aufstellen:
    - alle Zeilen mit Ergebnis 0 werden ausgewählt
    - wenn Eingangsvariable 1 □ negieren, ansonsten direkt übernehmen
    - Eingangsvariablen pro Zeile disjunktiv (=oder) verknüpfen
    - einzelne Zeilen konjunktiv (=und) verknüpfen
- **DNF = Disjunktive Normalform**
  - Vollkonjunktionen werden disjunktiv verknüpft „(x und y) oder (x und z)“
  - Wahrheitstabelle aufstellen:
    - alle Zeilen mit Ergebnis 1 werden ausgewählt
    - wenn Eingangsvariable 0 □ negieren, ansonsten direkt übernehmen
    - Eingangsvariablen pro Zeile konjunktiv (=und) verknüpfen
    - einzelne Zeilen disjunktiv (=oder) verknüpfen

Geben Sie jeweils ein typisches Beispiel pro Normalform mit drei booleschen Variablen a, b und c an, wobei jede Variable auch mindestens einmal in negierter Form vorkommen soll!

- **KNF:** (a und !b und c) oder (!a und b und !c)
- **DNF:** (a oder !b oder c) und (!a oder b oder !c)

Sie kennen Distributiv- und Kommutativgesetz sowohl vom Rechnen mit reellen Zahlen, als auch der booleschen Algebra. Wie lauten jeweils die zugehörigen Formeln? Gibt es Unterschiede zwischen der Verwendung in Arithmetik und boolescher Algebra (wenn ja, welche)?

- **Distributivgesetz:**
- $(a + b) * c = (a * c) + (b * c)\ für\ \forall a, b, c: \in R \rightarrow reelle\ Algebra$
- $a \cap (b \cup c) = (a \cap b) \cup (a \cap c) \rightarrow boolesche\ Algebra$

  $a \cup (b \cap c) = (a \cup b) \cap (a \cup c) \rightarrow boolesche\ Algebra$
- **Kommutativgesetz:**
- $a * b = b * a\ für\ \forall a, b: \in R \rightarrow reele\ Algebra$

  $a + b = b + a\ für\ \forall a, b: \in R \rightarrow reele\ Algebra$
- $a \cap b = b \cap a \rightarrow boolesche\ Algebra$

  $a \cup b = b \cup a \rightarrow boolesche\ Algebra$
- **Unterschiede:**
  - Die Verknüpfungen (+,*) beim Distributivgesetz mit reeller Algebra lassen sich nicht vertauschen wie bei der booleschen Algebra
- $a + (a * b) \neq (a + a) * (a + b)$

16.(I) Beweisen Sie mittels KV-Diagramm die De Morgansche Regel.

- !(a oder b) = !a und !b
-

a	b	!(a oder b)	!a und !b
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

	a	!a
b		
! b		x

- Aus dem KV – Diagramm ist ablesbar das die erste De Morgansche Regel bewiesen ist.

16.(I) Beweisen Sie mittels KV-Diagramm die De Morgansche Regel.

- !(a und b) = !a oder !b

a	b	!(a und b)	!a oder !b
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

	a	!a
b		x
!b	x	x

- Aus dem KV – Diagramm ist ablesbar das die minimale DNF = !a oder !b, wodurch auch die zweite De Morgansche Regel bewiesen ist.

17.(II) Wozu wird ein KV-Diagramm verwendet? Wie viele Felder hat ein KV-Diagramm mit n Eingangsvariablen und warum? Gibt es eine Obergrenze (upper limit) für die Anzahl der Eingangsvariablen? Begründen Sie Ihre Antwort?

- Graphische Veranschaulichung des Verfahrens von Quine und McClusky
- wird zum Vereinfachen von einer DNF verwendet und ermittelt die minimale DNF dieser
- es gibt 2^n Felder bei n Eingangsvariablen, weil man genau 2^n Möglichkeiten hat einen Ausdruck (z.B. !a und b und !c) zu formen
- ein KV – Diagramm ist für max. 4 Variablen sinnvoll anwendbar (ein Quadrat hat keine Seite für eine 5te Variable)

18.(II) **Gegeben ist folgender Term in minimaler DNF: a ∨ (b ∧ c).**
Tragen Sie diesen Term in ein KV Diagramm mit den Eingangsvariablen a, b und c ein, und markieren Sie alle Blöcke.

	a	a	!a	!a
b	x	x		x
!b	x	x		
	c	!c	!c	c

19.(II) Handelt es sich beim folgenden Satz um eine Tautologie, eine Antilogie (kontradiktion), oder keines von beiden? Erklären Sie ihre Antwort!

$$[(a \wedge \neg b) \rightarrow (a \vee c)] \vee (b \leftrightarrow c)$$

A	B	C	A und !B	□	A oder C	oder	B □□C
0	0	0	0	1	0	1	1
0	0	1	0	1	1	1	0
0	1	0	0	1	0	1	0
0	1	1	0	1	1	1	1
1	0	0	1	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	1	1	0
1	1	1	0	1	1	1	1

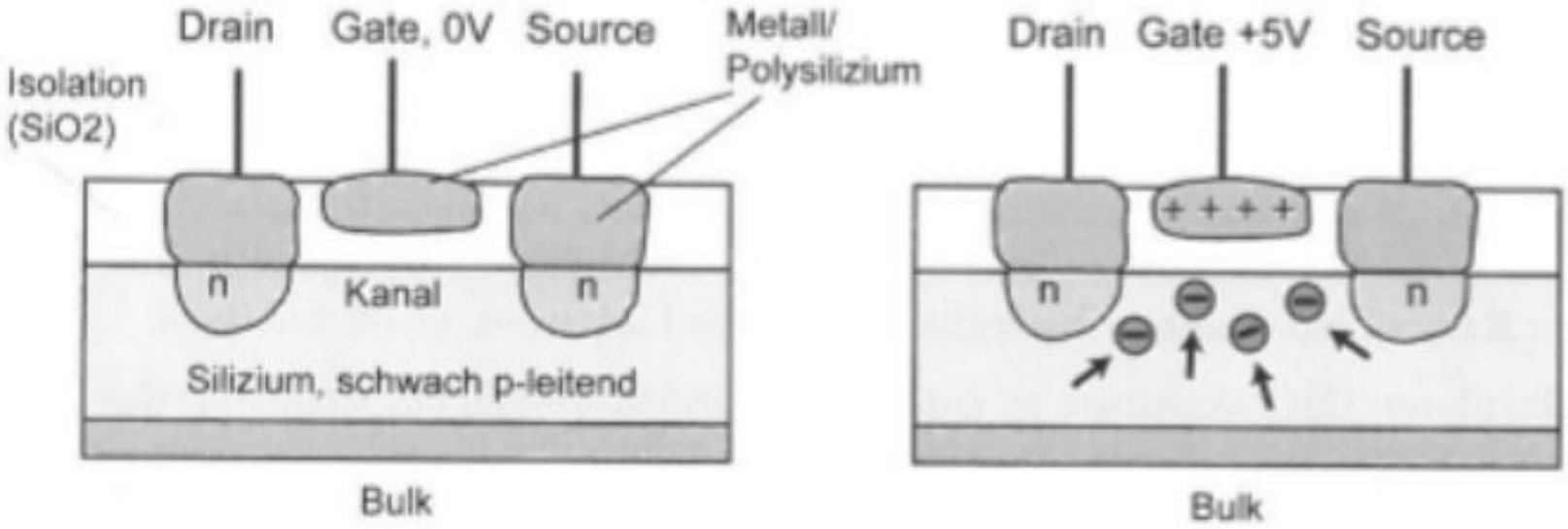
Tautologie,
weil der Ausdruck
immer 1 liefert

Weitere Prüfungsfragen
(Thema 3 – Digitale Logik)

Welche Anschlüsse hat ein n-Kanal-MOSFET, und wozu dienen diese Anschlüsse?

- **Gate (Tor, Gatter):** wenn nicht geladen, fließt kein Strom
- **Drain (Senke, Abfluss):** Strom fließt nach Source
- **Source (Quelle, Zufluss):** Strom kommt von Drain
- **Bulk (Substrat)**
- Da der NMOS ohne Ansteuerung gesperrt ist, nennt man ihn **selbstsperrend**

20. (I) **Skizieren Sie den Aufbau eine n-Kanal-MOSFET (NMOS) einschließlich der verschiedenen Dotierungen, und beschriften Sie alle Anschlüsse.** Erläutern Sie, auf welche Weise sich im NMOS ein leitender Kanal ausbilden kann.



- Ein leitender Kanal bildet sich erst aus, wenn positive Gate-Spannung anliegt (rechts)

- **Metal Oxide Semiconductor Field Effect Transistor (MOSFET):**
  - Vier Anschlüsse: Source – Drain – Gate – Bulk
  - Varianten:
    - n – Kanal – MOSFET (NMOS)
    - p – Kanal – MOSFET (PMOS)
- **NMOS**:
  - **Gate nicht geladen:** keine Ladungen im Kanal, kein Strom von Source nach Drain
  - **Gate geladen:** Kanal leitet
  - **Aber:** zusätzlich p-n-Übergang am Drain Sperrichtung
  - **Also:** NMOS leitet niedriges U_DS (Spannung_Drain_Source) falls hohes U_G (Spannung_Gate)
- **PMOS**:
  - Im Vergleich zum NMOS alles umgedreht!
  - Niedriges Potential am Gate:
- Kanal enthält Löcher, keine vollständige Rekombination
  - **Also:** PMOS leitet hohes U_DS (Spannung_Drain_Source) falls niedriges U_G (Spannung_Gate)

- **CMOS**:
  - Kombination von NMOS- (Pull-down-Pfad) und PMOS- (Pull-up-Pfad) Technologie auf gemeinsamen Substrat

21.(I) Wann bezeichnet man ein Gatter als „logisch vollständig“?

Nennen Sie zwei Beispiele für logisch vollständige Gatter

- **Bsp.** (NAND: liefert ‘0’ wenn alle Eingänge ‘1’ sind) ,
  (NOR: liefert ‘0’ wenn mindestens ein Eingang ‘1’ ist)

- Eine Gattermenge S wird als logisch vollständig bezeichnet, wenn alle Logikschaltungen damit erzeugt werden können. Die Vollständigkeit ist relativ einfach nachzuweisen, indem man die anderen Grundgatter aus den in S vorhandenen konstruiert.
- Im Grunde reicht es sogar aus, S0 := {AND, OR, NOT} aus S zu erzeugen, da S0 nachweislich logisch vollständig ist.

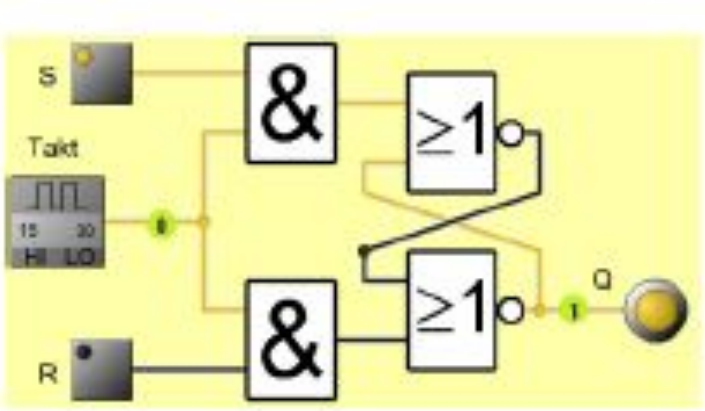
22.(II) Multiplexer: Wozu dienen die Steuerleitungen? Wie viele Steuerleitungen werden bei einem Multiplexer mit 8 Eingangsleitungen benötigt und warum?

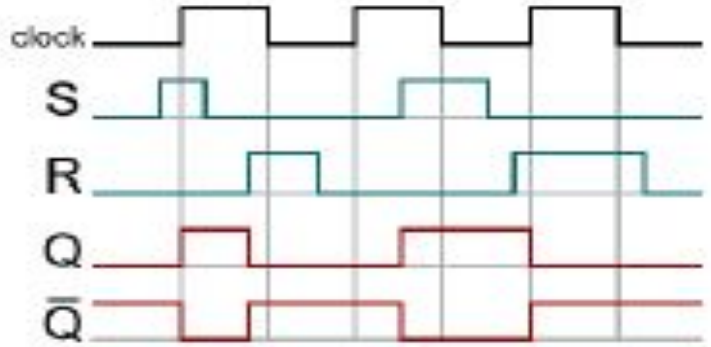
- Multiplexer wandelt Parallelen in seriellen Datenstrom um (durch Steuereingänge)
- Die Steuereingänge entscheiden welcher der Eingänge zum Ausgang geleitet wird -> ein. Datenausgang -> multiple input, single output!

- Bei 8 Eingangsleitungen werden mindestens 3 Steuerleitungen benötigt, weil:
- $n = \log_2(Eingängen)$

23.(II) Latches und Flip-Flops: Erklären Sie den Unterschied zwischen taktzustand-gesteurt und taktflanken-gesteurt.

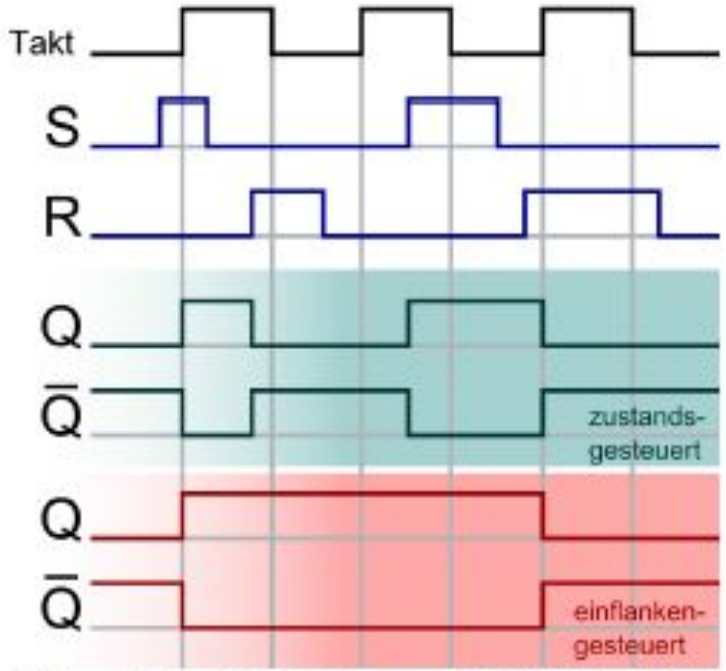
- **Latch ist taktzustands-(taktpegel-)gesteuert.**
  kann Zustand nur wechseln wenn Takt/Clock-Eingang aktiv (logisch 1) ist
- kann mehrmals während aktiver Taktphase den Zustand wechseln
- Zustand R = S = 1: weiterhin potentielle race condition





23.(II) Latches und Flip-Flops: Erklären Sie den Unterschied zwischen taktzustand-gesteurt und taktflanken-gesteurt.

- **FlipFlop ist taktflankengesteuert.**
- übernimmt den Eingangswert **NUR** zu dem Zeitpunkt, zu dem Takteingang von Low auf High wechselt
-



http://de.wikipedia.org/wiki/Flipflop

24.(I) Beschreiben Sie den Unterschied zwischen einem Decoder und einem Multiplexer? Welche Ein- und Ausgänge und welche zusätzlichen Leitungen gibt es jeweils?

Nennen Sie jeweils mindestens eine Anwendung!

- Decoder
  - **Multiple Input, Multiple Output**
  - Wandelt kodierten Input in kodierten Output um, wobei Input- und Output-Code unterschiedlich sind!
  - n-zu-2^n Decoder: n Eingänge, 2^n Ausgänge
  - Zu jeder Zeit ist nur ein Ausgang aktiv
  - Anwendung: **Instruction Decoder (CPU)**
    - Wandelt die Bits des Instruktionsregisters in Kontrollsignale um, die andere Teile der CPU steuern

24.(I) Beschreiben Sie den Unterschied zwischen einem Decoder und einem Multiplexer? Welche Ein- und Ausgänge und welche zusätzlichen Leitungen gibt es jeweils?

Nennen Sie jeweils mindestens eine Anwendung!

- Multiplexer
  - **Multiple Input, Singe Output**
  - der über die Steuereingänge gewählte Dateneingang wird unverändert zum Datenausgang geleitet
  - 2^n = m Dateneingänge
  - n = log2(m) = ld(m) Steuereingänge
  - Anwendung: **Tastatur**
    - jede Taste wird per 7-8 bit codiert, bei Anschlag werden die bits aber nicht parallel sondern seriell (hintereinander) über eine einzige Leitung übertragen

25.(I) Welche zwei wichtigen Vertreter von flüchtigen RAM Speichern kennen Sie (vollständige Bezeichnung!)

Erläutern Sie kurz ihre wesentlichen Eigenschaften und wofür sie verwendet werden!

- **SRAM – Statisches RAM**
  - schnelle Lesezugriffe und Umschaltzeiten
  - kein Refresh nötig, **dennoch flüchtig**
  - teurer, größer als DRAM
  - besteht aus **taktgesteuerten** Flip-Flops
  - (meist 6 Transistoren)
  - für Register, Akkumulatoren
  - und Caches verwendet
-

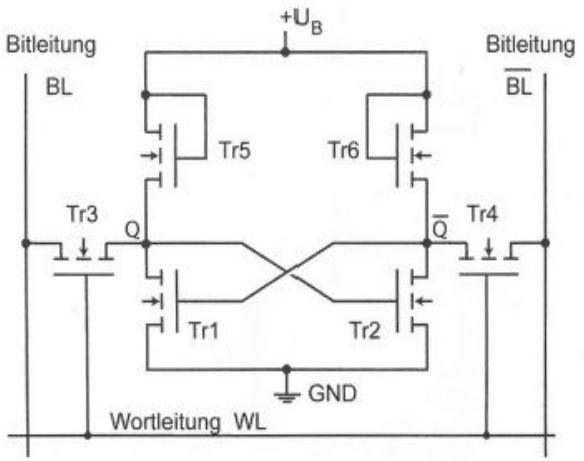


Abbildung 4.13: Das Flipflop aus zwei NMOS-Transistoren ist der Kern der SRAM-Speicherzelle. Die Wortleitung schaltet über zwei weitere Transistoren die Ein-/Ausgänge auf die Bitleitungen BL und $\overline{BL}$ durch. Über diese erfolgt das Schreiben und Lesen von Daten.

25.(I) Welche zwei wichtigen Vertreter von flüchtigen RAM Speichern kennen Sie (vollständige Bezeichnung!)

Erläutern Sie kurz ihre wesentlichen Eigenschaften und wofür sie verwendet werden!

- **DRAM – Dynamischer RAM**
  - besteht aus Kondensator und Transistor
  - regelmäßiger Refresh nötig, sonst Datenverlust (32ms oder 64ms)
  - kleiner und billiger als SRAM Hauptspeicher
  - in Array angeordnet – immer ganze Zeile aktiv

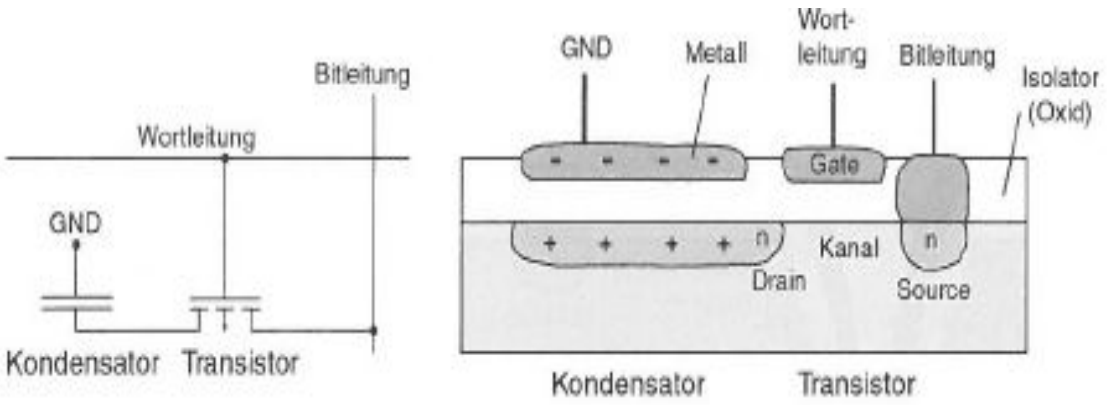


Abbildung 4.14: Die Zelle eines DRAM besteht aus einem Kondensator und einem Tortransistor. Links Schaltbild, rechts Schichtenaufbau.

Erklären Sie die grundlegenden Unterschiede zwischen den Speichertypen RAM und ROM! Wofür stehen die Abkürzungen?

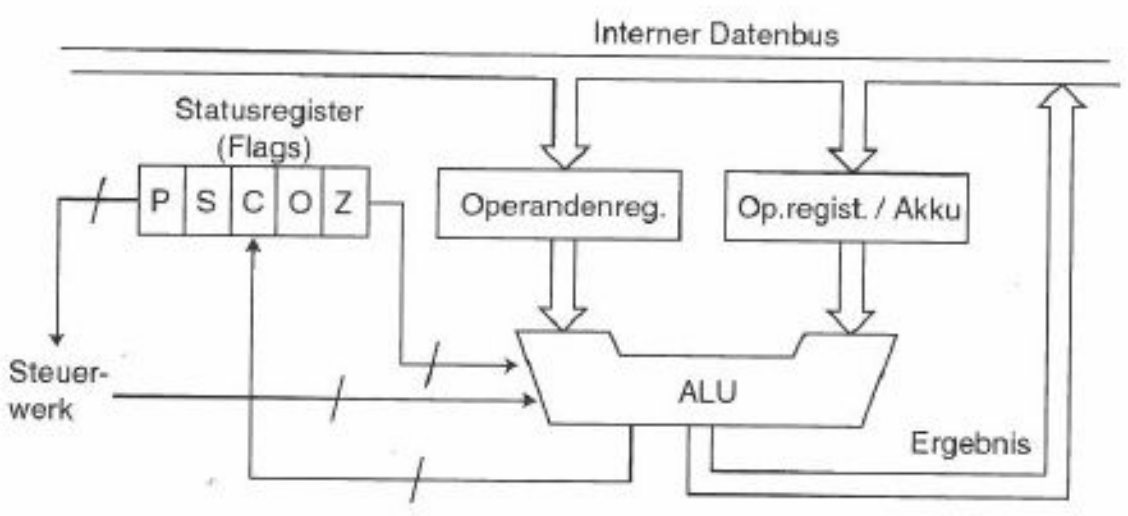
- **RAM = Random Access Memory**
  - verwendet bei Hauptspeicher, Arbeitsspeicher, usw.
  - es kann in beliebiger Reihenfolge zugegriffen werden
  - Anwendungen und Daten die gerade bearbeitet oder gelesen werden, werden dazu in den Speicher geladen
  - **flüchtiger Speicher**
- **ROM = Read Only Memory**
  - verwendet bei BIOS, Messgeräte, usw.
  - kann im normalen Betrieb nur gelesen werden
  - kommen zum Einsatz um Daten dauerhaft und unabänderlich zu speichern
  - **nicht flüchtiger Speicher**

Prozessor: Was versteht man unter Register bzw. Registersatz? Erklären Sie die grundlegenden Eigenschaften von Registern

- **Registersatz:** Nach außen „sichtbare“ Register. Er besteht aus:
  - **Universalregister**: für verschiedene Zwecke verwendbar
  - **Adressregister**: Speichern Speicheradressen eines Operanden oder Befehls
  - **Spezialregistern**: für bestimmte Zwecke vorgesehen
    - Befehlszählregister: Speicheradresse des nächsten Befehls
    - Befehlsregister: aktueller Befehl
    - Statusregister: z.B. auftreten eines Überlaufs
- **Register:**
  - ein Speicherbereich innerhalb eines Prozessors
  - durch internen Datenbus meist direkt mit ALU verbunden
  - schneller und kleiner als Hauptspeicher
  - gewisse Breite, meist 8,16,32,64…Bit (besteht aus Flip-Flops, 1 Bit je)

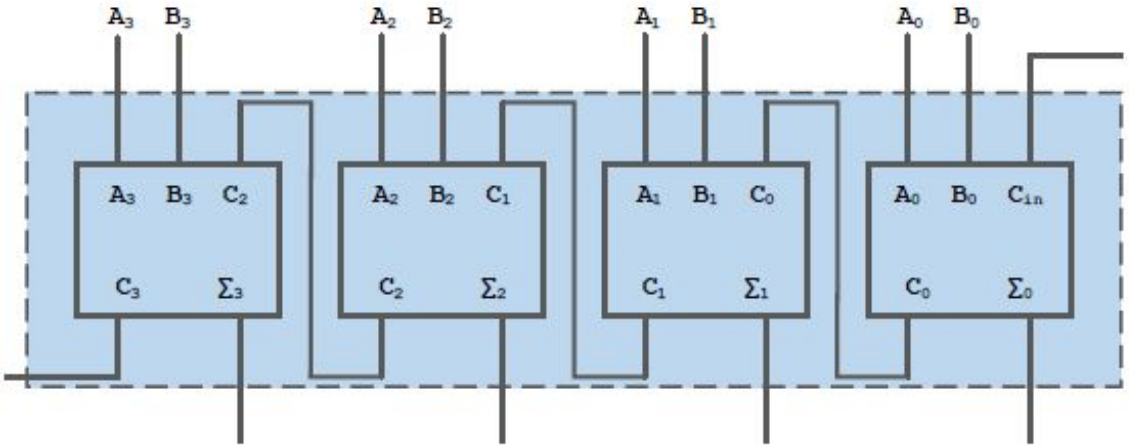
Prozessor: Wofür steht die Abkürzung ALU und welche Aufgaben hat die ALU?

- **ALU = Arithmetical Logical Unit**
  - führt die vom Steuerwerk verlangten **logischen und arithmetischen** Operationen aus
  - wird vom Steuerwerk nach Dekodierung einer entsprechenden Instruktion angesprochen
  - keine eigenen Speicherzellen: es werden Operandenregister auf die Dateingänge für die Zeit der Berechnung aufgeschaltet



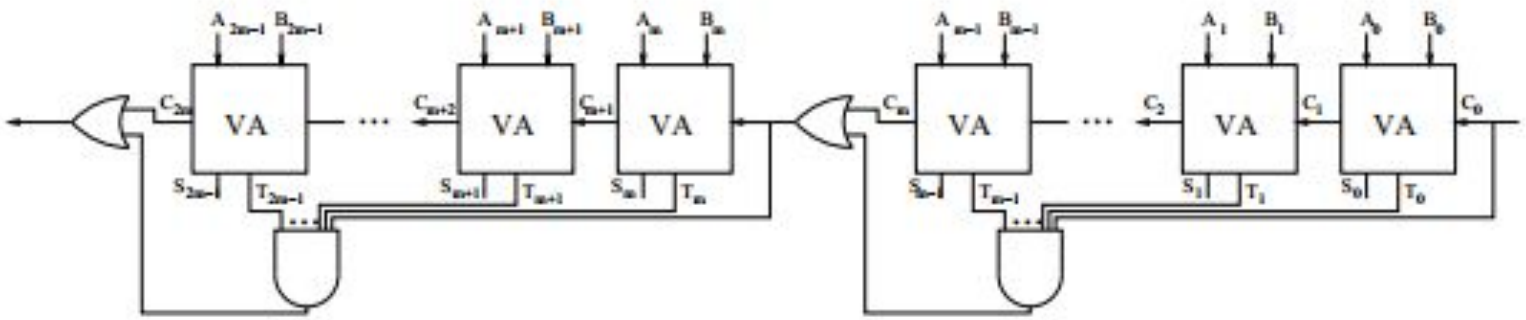
26.(I) Carry-Ripple-Addierer: Erklären Sie das Prinzip des Carry-Ripple-Addierers! Wie viele Binärzahlen kann er addieren? Wie viele Stellen können diese Binärzahlen haben?

- kann mit Volladdierer oder Halbadderier beginnen
- 2 **n-stellige** Binärzahlen addieren (A_n bzw. B_n = msb; A_0 bzw. B_0 = lsb)
- Übertrag der jeweils niederwertigeren Stelle muss berücksichtigt werden
- **Nachteil:**
  - jeder Addierer braucht vorher das C_Out des vorherigen Addierers
  - im **Worst – Case** muss das Carry – Bit die ganze Schaltung durchlaufen (**carry propagation**)



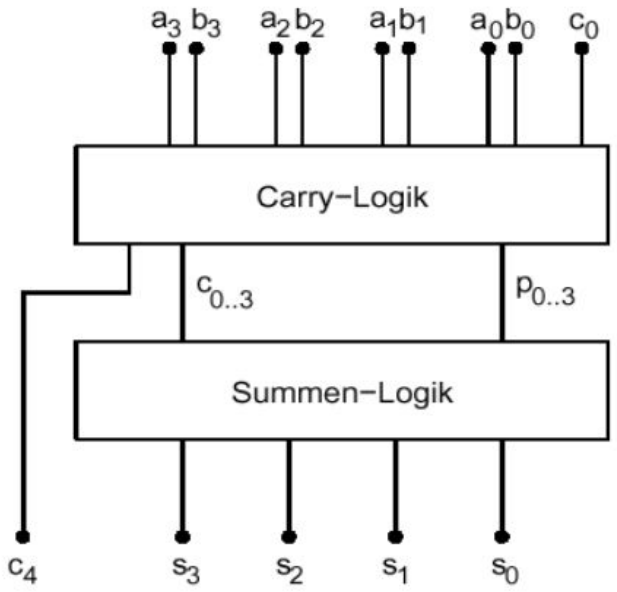
26.(I) Carry-Ripple-Addierer: Erklären Sie das Prinzip des Carry-Ripple-Addierers! Wie viele Binärzahlen kann er addieren? Wie viele Stellen können diese Binärzahlen haben?

- **Lösungen:**
  - **Carry – Skip – Addierer**
    - Volladdierer werden gruppiert
    - Zusatzlogik untersucht, ob sich ein Übertrag durch gesamt Gruppe propagiert
    - wenn ja, wird dies der nächsten Gruppe gemeldet, damit diese ebenfalls mit der Berechnung anfangen kann



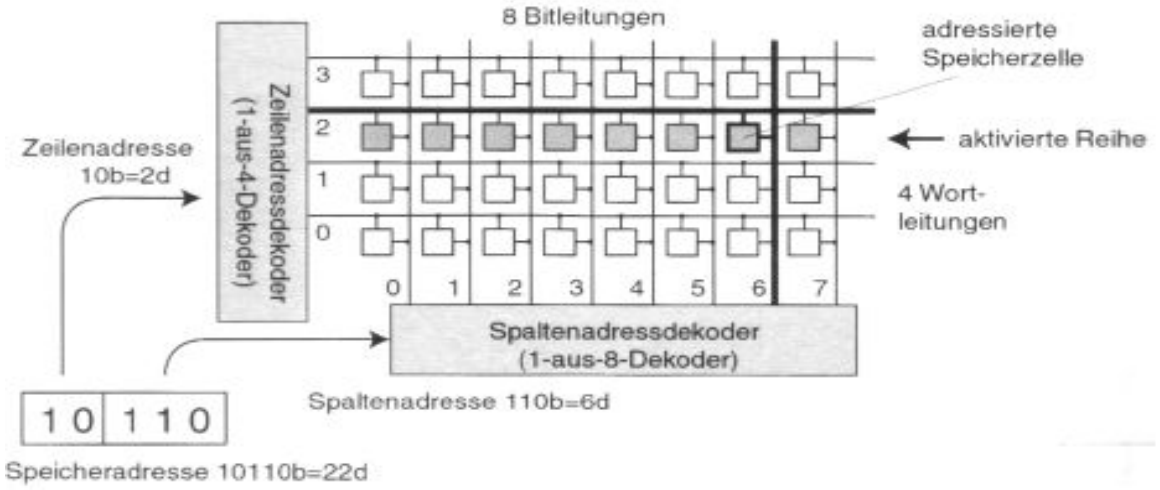
26.(I) Carry-Ripple-Addierer: Erklären Sie das Prinzip des Carry-Ripple-Addierers! Wie viele Binärzahlen kann er addieren? Wie viele Stellen können diese Binärzahlen haben?

– **Carry – Look – Ahead – Addierer**
  - Überträge werden bereits im ersten Additionsschritt ermittelt
  - größerer Schaltungsaufwand nötig
  - nur für kleine Wortbreiten (länge der Binärzahlen) effektuv



Skizzieren Sie den typischen Aufbau eines 32 Bit Speicherbausteines mit Wortleitungen und 8 Bitleitungen. Erklären Sie, wie die Speicheradresse 01|101 geschrieben bzw. gelesen werden kann.

- Speicheradresse wird auf 01 und 101 aufgeteilt
- 01 geht an den Zeilenadressdecoder
- die Wortleitung 1 wird aktiviert
- 101 geht an den Spaltenadressdecoder
- die Bitleitung 5 wird aktiviert
- nur das Signal dieser Bitleitung wird gelesen/geschrieben



27.(II) **Wie viele Binärzahlen kann ein Volladdierer addieren und wie viele Stellen können diese Binärzahlen haben? Wie viele Binärzahlen kann ein Carry-Ripple Addierer addieren und wie viele Stellen können diese Binärzahlen haben?**

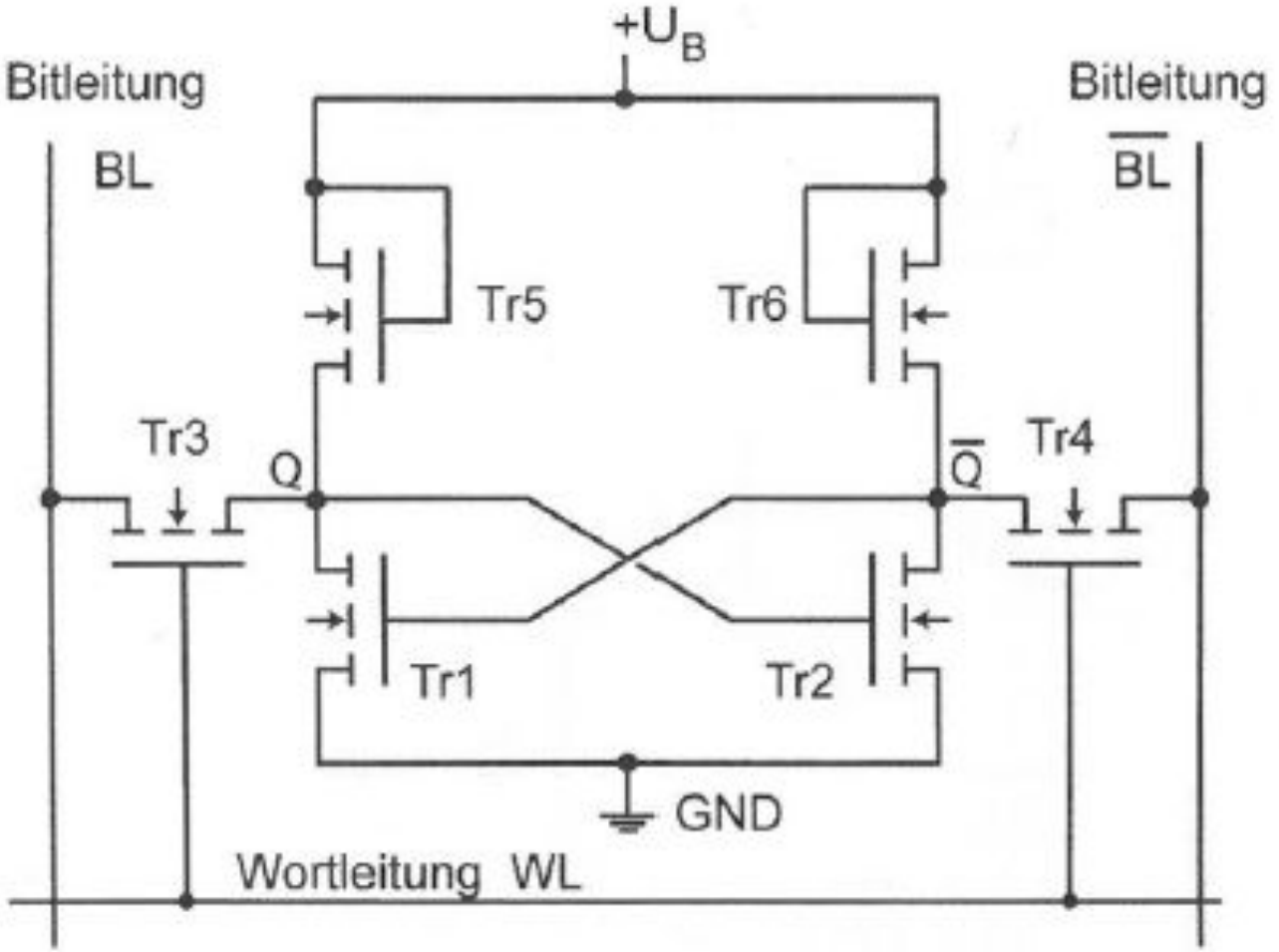
- **Volladdierer:**
- Ein 1 bit Volladdierer kann 3 einstellige Binärzahlen addieren (E1, E2, C_IN)
- **Carry – Ripple – Addierer:**
- 2 **n-stellige** Binärzahlen können addiert werden

28.(II)Geben Sie die grundlegenden Eigenschaften von Flash-Speichern an! Zu welcher Familie (RAM/ROM) von Halbaleiterspeichern (engl. Semiconductor) gehören Flash-Speicher?

- Flash – Speicher ist eine **spezielle EEPROM – Variante**
- Löschen/Beschreiben der Zellen erfordert geringere Spannungen/Ströme
- **nur ganze Blöcke von Zellen löschbar/beschreibbar**
- arbeitet ähnlich wie **RAM**-Baustein, aber **nicht – flüchtig**
- z.B.: SSD, USB – Memory – Stick, usw.

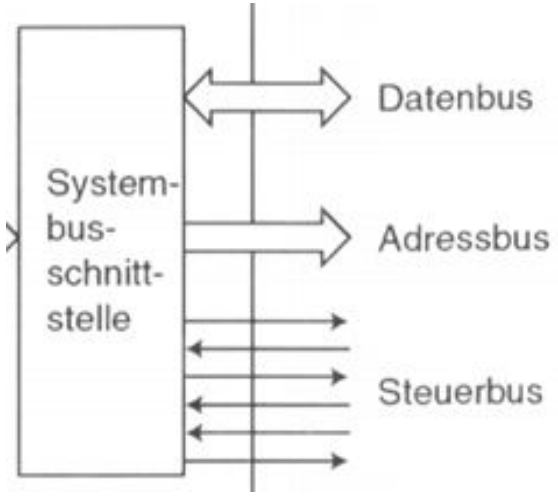
29.(II) **Aus wie vielen Transistoren besteht eine SRAM-Speicherzelle typischerweise? Skizieren Sie den grundsätzlichen Aufbau einer SRAM-Zelle!** (Vergessen Sie nicht auf Wort- und Bit-Leitungen)

- SRAM-Zelle besteht meist aus 6 Transistoren (s.371/373)



Prozessor: Welche Aufgabe haben Busse? Über welche Busse ist die CPU an den Rest des Systems gebunden?

- sind interne Kommunikationsleitungen
- über Busse können verschiedene interne Systeme miteinander verbunden werden
- Die CPU ist über den **Systembus** an das System angeschlossen, Weiterverbindung über:
  - Datenbus
  - Adressbus
  - Steuerbus



Paging: Erklären Sie den Unterschied zwischen physikalischem und virtuellem Adressraum.

- physikalischer Adressraum:
  - befindet sich vollständig am physikalischem Speicher (RAM)
- virtuelle Adressraum:
  - ist unabhängig vom physikalisch vorhandenen Speicher
  - wird meist auf Festplatte in einer sogenannten Page Datei ausgelagert
  - Programme behandeln den Adressraum gleich wie den RAM
  - Organisation übernimmt das Betriebssystem

Caches: Warum wird zwischen MR und MPI (nicht MP!) unterschieden?

- MR = Miss Rate = Misses/Speicherzugriff
  - spiegelt die Misses für **alle** Speicherzugriffe wieder
- MPI = Misses/Instruktion
  - spiegelt die Misses pro Instruktion wieder
  - in einer Instruktion kann es **mehrere** Speicherzugriffe geben

- **Ein Prozess kann nur von einem anderen Prozess beendet werden.**
- **Falsch –** in kann auch durch das Betriebssystem (System Call), durch sich selbst (Aufgabe erfüllt) oder durch höhere Macht (Stromausfall) beendet werden

- **Der Benutzer ist für den Speicherschutz zwischen Prozessen verantwortlich.**
- **Falsch –** bei der Prozesserzeugung wird automatisch ein eigener Adressraum für den Prozess angelegt, der vor anderen Prozessen geschützt ist

- **Ein Thread ist ein Programm in Ausführung.**
- **Falsch –** ein Prozess ist ein Programm in Ausführung und dieser Prozess kann mehrere Threads enthalten

- **Viele Context Swichtes in kurzer Zeit können einen negativen Einfluss auf die Performance eines Systems haben.**
- **Richtig –** Context Swichtes brauchen eine gewisse Zeit, da der Zustand von P1 gespeichert und der Zustand von P2 geladen werden muss, bei vielen kann es zu einem zu großen Overhead kommen

Beim CPU Scheduling kann die durchschnittliche Wartezeit dadurch minimiert werden, dass Prozesse mit langer Ausführungszeit zuerst ausgeführt werden.
Falsch – Dadurch würde die Wartezeit für spätere Prozesse extrem steigen. Es ist genau umgekehrt, Prozesse mit kurzer Ausführungszeit werden zuerst ausgeführt

In interaktiven Betriebssystem gibt es üblicherweise weniger Context Switches als in Stapelverarbeitungssystemen.
Falsch – interaktive Betriebssysteme haben üblicherweise mehr Context Switches als Stapelverarbeitungssysteme, da sie versuchen alle Prozesse möglichst gleichzeitig abzuarbeiten

Round Robin Scheduling minimiert die durchschnittliche Wartezeit pro Prozess.
Richtig – jeder Prozess ist maximal für ein Quantum aktiv bis ihm die CPU wieder genommen wird, dadurch wird die CPU auf alle Prozesse fair aufgeteilt. Jedoch ist fraglich ob wirklich das Minimum an Wartezeit erreicht wird.

30.(I) Erklären Sie in wenigen Worten das Prinzip des virtuellen Speichers

- wird bei Speichermangel im Hauptspeicher (RAM) eingesetzt
- wird von der Memory Management Unit (MMU) auf der CPU geregelt
- virtueller Adressraum ist unabhängig vom physikalisch vorhandenen Arbeitsspeicher
- virtuelle Adresse wird zu einer physikalischen Adresse
- bei Speichermangel werden Bereiche in Massenspeicher ausgelagert (Paging)

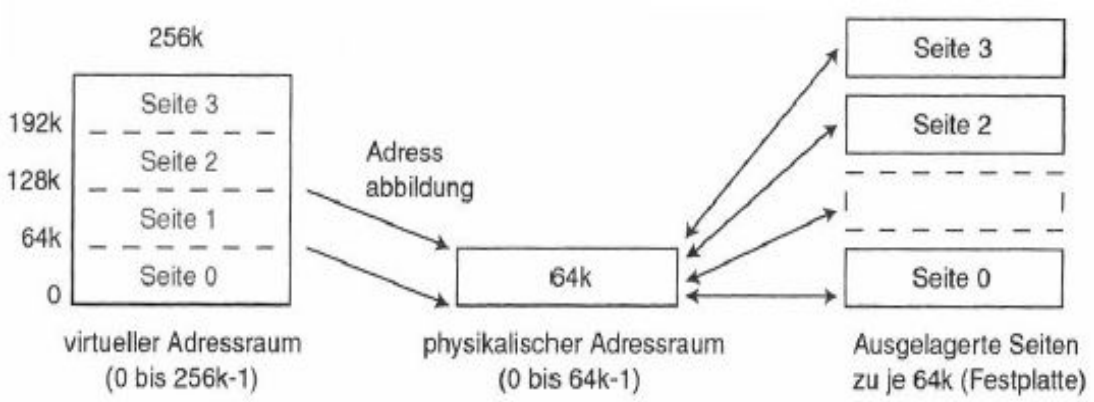


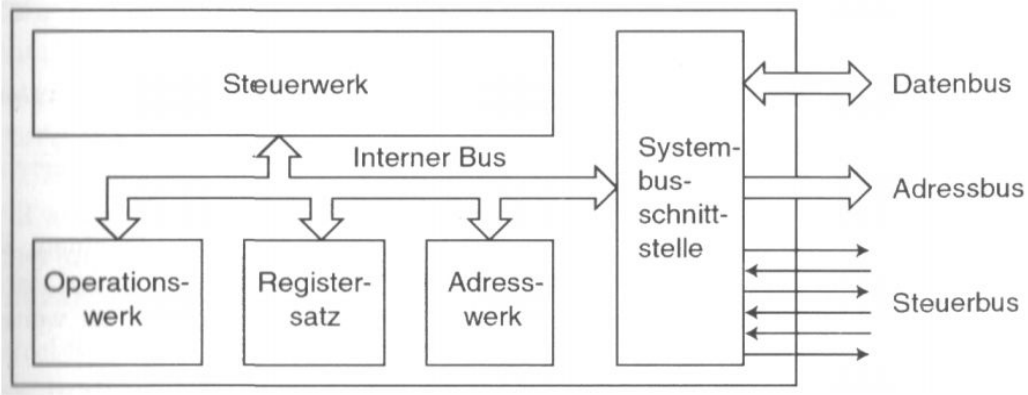
Abbildung 10.1: Die Seitenauslagerung ermöglicht einen beliebig großen virtuellen Adressraum. In diesem Beispiel befindet sich gerade Seite 1 im physikalischen Speicher, die anderen Seiten sind ausgelagert.

31.(I) Erklären Sie, warum man in CPUs schon seit langer Zeit Caches verwendet.

- CPU wurde in größerem Maße schneller, als der Hauptspeicher
- teilweise dauert der Speicherzugriff oft mehr als 10 CPU-Takte
- **CPU – Designziel**: pro Takt einen Befehl abarbeiten
- Daten und Code müssen schnell genug aus Speicher geliefert werden, ansonsten wäre der Fortschritt bei den CPU´s sinnlos
- deswegen wurden **Caches** (kleine, schnelle Zwischenspeicher) eingeführt

32.(II) Aus welchen grundlegenden Komponenten besteht eine CPU? Beschreiben Sie jede dieser Komponenten mit einem Satz.

- **Registersatz**: Register, um Daten innerhalb des Prozessors speichern zu können
- **Steuerwerk**: Verantwortlich für Ablaufsteuerung
- **Operationswerk** (ALU): Eigentliche Datenverarbeitung
- **Adresswerk:** Um auf Daten und Code im Hauptspeicher zugreifen zu können
- **Systembus –** Schnittstelle: Datenverkehr mit Rest des Systems



33.(II) Beschreiben Sie kurz (aufzählen reicht nicht) mindestens zwei Probleme die beim Pipelining auftreten können und diskutieren Sie pro Problem mindestens eine Lösung.

Takt	Instr-Fetch	Decode	Opnd-Fetch	Execute	Write
1	Befehl 1	×	×	×	×
2	Befehl 2	Befehl 1	×	×	×
3	Befehl 3	Befehl 2	Befehl 1	×	×
4	Befehl 4	Befehl 3	Befehl 2	Befehl 1	×
5	Befehl 5	Befehl 4	Befehl 3	Befehl 2	Befehl 1
6	Befehl 6	Befehl 5	Befehl 4	Befehl 3	Befehl 2
7	Befehl 7	Befehl 6	Befehl 5	Befehl 4	Befehl 3
8	Befehl 8	Befehl 7	Befehl 6	Befehl 5	Befehl 4

- Ziel ist Zusammenlegung mehrerer Befehle zur gleichen Zeit. In jedem Takt:
  - Fertigen Befehl herausnehmen
  - Jeden anderen Befehl in nächste Bearbeitungsstufe
  - Neuen Befehl in erste Stufe
- Takt 1-5: Latenzzeit nur beim Befüllen relevant
- Takt 6-8: Ein Befehl pro Takt
- Nach befüllen theoretisch 100% Auslastung möglich

33.(II) Beschreiben Sie kurz (aufzählen reicht nicht) mindestens zwei Probleme die beim Pipelining auftreten können und diskutieren Sie pro Problem mindestens eine Lösung.

- **Probleme:**
  - **Interlocks:**
    - Hauptspeicherzugriff, kann nicht immer in einem Takt erfolgen
    - Nur dann, wenn Datum in L1 – Cache und dieser in vollem CPU – Takt betrieben
  - **Data Hazards / Read – After – Write Hazard (Datenabhängigkeit):**
    - Befehl 1 schreibt in Register das von Befehl 2 gelesen werden muss
    - Befehl 2 muss warten, bis Befehl 2 fertig
    - **Lösung:**
      - Leerbefehl (Leistungseinbußen),
      - Lösung über Compiler (Datenabhängigkeiten erkennen und gegebenenfalls Instruktionen umordnen)
  - **Bedingte Sprungbefehle:**

33.(II) Beschreiben Sie kurz (aufzählen reicht nicht) mindestens zwei Probleme die beim Pipelining auftreten können und diskutieren Sie pro Problem mindestens eine Lösung.

- **Probleme:**
  - **Bedingte Sprungbefehle:**
    - Programmteile können übersprungen werden
    - Sprungziel (Program Counter!) erst nach Auswertung (Ausführung) bekannt
    - **Lösung:**
      - Branch Prediction (Annahme ob Sprung genommen), wenn richtig kein Zeitverlust
      - Branch History Table:
        - » Sprungbefehle im aktuellen Programmablauf beobachten und Statistik aufstellen
        - » sehr aufwändig 99% Trefferquote

34.(I) Warum ist Pipelining für die Performance einer CPU wichtig? Diskutieren Sie mindestens zwei Problem beim Pipelining auftreten können? Welche Lösungsmöglichkeiten gibt es?

- Siehe frage 33

35.(I) Erklären Sie die beiden Begriffe CISC und RISC. Worin liegen die Unterschiede zwischen diesen beiden Architekturen?

- **CISC, Complex Instruction Set Computer**
  - viele, verhältnismäßig mächtige Einzelbefehle
  - **Mikroprogrammierung:** Sequenzen für Steuerung der CPU werden aus Mikrocode – ROM abgerufen
  - Befehlssatz wurde immer größer und immer kompliziertere Befehle
  - **Vorteile:**
    - auf Software – Ebene erweiterbar
    - Fehlerbehebung: Neuer Mikrocode auch beim Kunden einspielbar
    - **Kompatibilität** – Emulation: Befehlssatz von Vorgängern auf SW – Ebene nachbildbar
  - **Nachteile:**
    - Dekodierung der vielen komplexen Befehle sehr aufwändig
    - Dekodierungseinheit brauchte mehr Zeit und Platz auf Chip

35.(I) Erklären Sie die beiden Begriffe CISC und RISC. Worin liegen die Unterschiede zwischen diesen beiden Architekturen?

- **RISC, Reduced Instruction Set Computer**
  - kein Mikrocode, keine algorithmische Abarbeitung
  - Befehl **muss** in Hardware implementiert sein
  - enthält keine komplizierten Befehle
  - **Ziel:** Möglichst in jedem Takt einen Befehl bearbeiten (Skalarität; >1 □ superskalar)
- Computer heute haben meist eine Mischung aus beiden Architekturen

36.(I) **Erklären sie, warum bei mehrstufigen Caches die Miss Penalty (MP) des Level 1 (L1) Caches wie folgt berechnet werden kann: MP(L1) = HitTime(L2) + MR(L2) * MP(L2)** (3P)

- wenn die Daten im L1 Cache nicht gefunden werden,
  wird im L2 Cache gesucht

- daher setzt sich die Miss Penalty (MP) von L1,
  aus der HitTime auf den L2 + der MP vom L2,

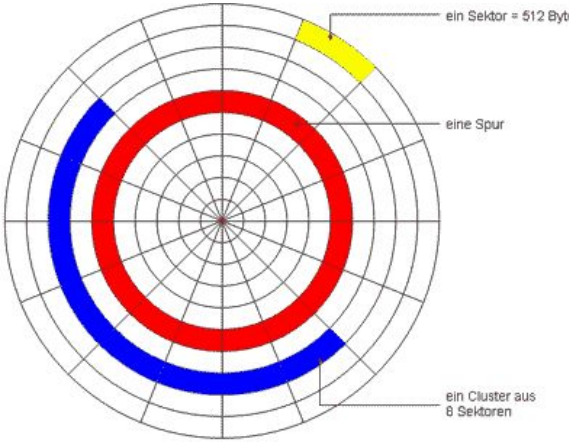
die allerdings nur bei einem Cache Miss auftritt

und deswegen mit der Miss Rate vom L2 multipliziert wird

37.(I) (Magnetische) Festplatte:

Warum können defragmentierte Daten schneller gelesen werden als stark fragmentierte Daten?

Welche(r) Teil(e) der Gesamttrasferzeit (T_a = T_s + T_r + T) sind betroffen? (2P)



Defragmentiert:

- logisch zusammengehörende Daten liegen nebeneinander
- Lesekopf muss im Idealfall nur einmal an die richtige Position gebracht werden

Fragmentiert:

- logisch zusammengehörende Daten, sind auf der ganzen Festplatte verteilt
- Lesekopf muss **n-mal** an die richtige Position gebracht werden
- **T_a = n * (T_s + T_r) + T**

38.(II) Caches: Was besagt das Lokalitätsprinzip? Erklären Sie kurz die zwei Arten von Lokalität?

- auf (Haupt-)Speicher wird meist nicht völlig zufällig zugegriffen
- 2 Prinzipien:
  - **Räumliche Lokalität:** Häufig Zugriffe auf Adressen, die in der Nähe kürzlich benutzter Adressen liegen
  - **Zeitliche Lokalität:** Folgezugriffe auf kürzlich benutzte Adresse
- **Ziel:** kürzlich benutzte Daten möglichst lange im Cache halten
- Nach Hauptspeicherzugriff wird nicht nur Inhalt der adressierten Speicherzelle im Cache aufbewahrt (**zeitliche Lokalität**), sondern gleich der ganze Speicherblock (**räumliche Lokalität**), in dem die Speicherzelle liegt

39.(II) Caches: Erklären Sie die Begriffe: Hit Time, Miss Rate, Miss Penalty

- tH … Hit Time, Zeit um Treffer im Cache zu erzielen
- MR … Miss Rate, Wahrscheinlichkeit gesuchtes Datum nicht zu finden
- MP … Miss Penalty, Extra – Aufwand wenn Datum nicht gefunden wird
- $Mittlere\ Speicherzugriffszeit\ (ein\ Cache)\ t_{SZ} = tH + MR * MP$

Weitere Prüfungsfragen
(Thema 5 –Betriebsysteme)

40.(II) Erklären Sie das Konzept des Multiprogramming und warum es vorteilhaft für die Auslastung einer CPU ist.

- Wenn ein Job auf I/O wartet, wird ein anderer Jobausgeführt Speicherschutz etc. notwendig
- es entsteht eine Konkurrenz um die CPU Zeit innerhalb der Jobs

41.(II) Erklären Sie die wichtigsten Unterschiede zwischen Prozessen und Threads bezüglich Ressourcenverwaltung, Ausführungsverwaltung, Speicherschutz, und Effizienz

- **Prozess** = Programm in Ausführung
  - jeder Prozess ist einem Adressraum zugeordnet
    - besteht aus Liste von Speicherstellen
    - beinhaltet: ausführbares Programm, Programmdaten, etc.
  - jedem Prozess ist eine Ressourcenmenge zugeteilt
    - Register, geöffnete Dateien, Liste verbundener Prozessoren etc.

- **Thread** = Art Prozess innerhalb eines Prozesses, jedoch
  - leichtgewichtiger als Prozesse: ca. 10-100x schneller zu erzeugen und zu zerstören, dafür schwieriger zu implementieren
  - **ohne** eigenem Adressraum (teilen sich Ressourcen, kein Speicherschutz)
  - **selbe** Zustände wie Prozesse

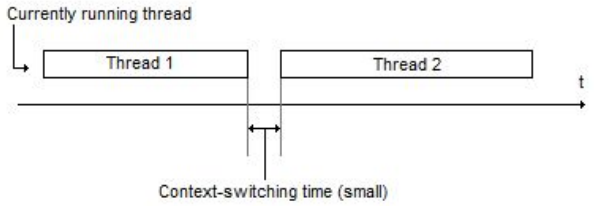
41.(II) Erklären Sie die wichtigsten Unterschiede zwischen Prozessen und Threads bezüglich Ressourcenverwaltung, Ausführungsverwaltung, Speicherschutz, und Effizienz

- **Unterschiede**
  - Prozess Modell: Bündelung von Ressourcen & Ausführung
  - bei Threads ist das aufgetrennt
  - **Ressourcenverwaltung:** Threads teilen sich die Ressourcen; kein Speicherschutz
  - **Ausführungsverwaltung:** Threads erweitern das Prozess Modell um die Möglichkeit, mehrere Ausführungsfäden, die sich in hohem Grade unabhängig voneinander verhalten, in derselben Prozessumgebung laufen zu lassen

Elemente pro Prozess	Elemente pro Thread
Adressraum	Befehlszähler
Globale Variable	Register
Geöffnete Dateien	Stack
Kindprozesse	Zustand
Ausstehende Signale	
Signale und Signalroutinen	
Verwaltungsinformationen	

42.(II) Wann spricht man von einem Context Switch? Warum können viele Context Switches in kurzer Zeit einen negativen Einfluss auf die Performance haben?(Tipp: siehe s.738)

- ein context switch (Prozesswechsel) tritt dann auf, wenn ein noch **nicht fertiger**, rechnender Prozess A die CPU freigeben muss, damit ein anderer Prozess B diese verwenden kann
- ein context switch inkludiert:
  - die Speicherung des Zustandes von Prozess A
  - das Laden des Zustandes von Prozess B
- Laden und Speichern erfordert einen gewissen Zeitaufwand
-
-
-
-



- wie in der Abb. zu erkennen ist dauert ein Context Switch normalerweise kürzer als die Ausführungszeit eines Prozesses/Thread. Kommt es jedoch zu vielen Context Switches in kurzer Zeit, kann es passieren, das die Zeit für Context Switches größer ist, als die Zeit in der ein Prozess/Thread ausgeführt wird.

43.(II) Diskutieren Sie mindestens drei mögliche Gründe wann eine CPU-Scheduling Entscheidung getroffen werden kann/muss?

- Neuer (Kind-)Prozess wurde erzeugt
- (Gerade rechnender) Prozess wurde beendet
- (Gerade rechnender) Prozess ist blockiert, z.B. wegen I/O
- Interrupt von I/O – Gerät (z.B. Transfer fertig)
- Prozess benutzt CPU schon für einen gewissen Zeitraum
- **Ziele**
  - **Fairness**: Jeder Prozess bekommt CPU – Zeit
  - **Policy Enforcement:** Vorgegebene Strategien durchsetzen
  - **Balance:** Alle Systemteile möglichst gut auslasten

44.(II) Erklären Sie präzise die preemptive Priority Scheduling Strategie mit Aging. Warum ist das Prinzip des ‚Aging' sinnvoll?

- **Non-preemptive Scheduling:**
  - Prozess kann solange die CPU benutzen bis er blockiert oder bis er freiwillig die CPU freigibt
- **Preemptive Scheduling:**
  - Prozess darf CPU nur für bestimmte Zeit beanspruchen
  - nach dieser Zeit wird der Prozess unterbrochen (egal ob fertig oder nicht)
  - anderer Prozess wird ausgeführt, fall rechenbereit
  - erfordert Taktgeber
- **Priority Schedulung:**
  - Prozess wird Priorität zugeordnet
  - Prozess mit höchster Priorität (bei Aging = 0) wird ausgeführt
- **Aging:**
  - Priorität des laufenden Prozesse wird zwecks Fairness üblicherweise verringert

Thema 5 - Betriebsysteme
JA/NEIN Fragen + Begründung

Betriebsysteme JA/NEIN Fragen

1. Betriebssysteme verwalten zwar den Zugriff auf die CPU; Betriebssysteme verwalten jedoch ansonsten keine weiteren Betriebsmittel

Falsch - Betriebssysteme verwalten auch ein und Ausgabegeräte

verwalten Speicher (Register, Caches, Ram, Disk, usw.).. also insgesamt CPU Speicher

2. Betriebssysteme ermöglichen zwar die Verwendung unterschiedlicher Hardware, dadurch wird die Kommunikation zwischen Anwendungsprogrammen und der darunter liegenden Hardware jedoch meist noch komplizierter.

Falsch- Treiber vereinfachen die Kommunikation

3. Bei einem Deadlock versucht ein Prozess einen anderen Prozess zu terminieren (zu töten)

Falsch-Deadlock entsteht wenn ein Prozess auf Ressourcen zu greifen will diese jedoch von einem anderen Prozess verwendet werden.

4. Bei Multiprogramming können mehrere Prozesse gleichzeitig auf derselben CPU (derselben ALU) rechnen.

Falsch-Prozesse können nur nacheinander oder zwischendurch bearbeitet werden nicht gleichzeitig

5. Alle Threads innerhalb eines Prozesses haben einen unterschiedlichen Adressrau,.

Falsch- Sie teilen sich den Adressraum des Prozesses

Betriebsysteme JA/NEIN Fragen

6. Jeder Thread hat seinen eigenen Befehlszähler und einen eigenen Zuständ.

Richtig- Da jeder Thread für sich selbständig arbeitet

7. Das Betriebssystem ist für den Speicherschutz zwischen Threads desselben Prozesses verantwortlich.

Falsch- Für den schutz der Threads ist der Programmierer zuständig

8. Ein sehr kurzes Quantum beim Round-Robin Scheduling ist immer besser als ein langes Quantum

Falsch-ist die zeit zu kurz kommt es zu vielen Context switsches

9. Die Ausführung von Systemaufrufen (zB.: read, write, ...) wird vom Compiler gesteuert.

Falsch- der Compailer übersetzt ein Programm A der Sprache Ai in ein Programm B der Sprache Bi --- es wird von Betriebssystem gesteuert.

10. Ein context switch hat keinen negativen Einfluss auf die Performance eines Systems.

Falsch- Der Context zwischen muss den alten Prozess speicher bevor er den neuen ausführt

Betriebsysteme JA/NEIN Fragen

11. Beim CPU Scheduling kann die durchschnittliche Wartezeit dadurch minimiert werden, dass Prozesse mit kurzer Ausführungszeit zuerst ausgeführt werden.

Richtig – Wenn die Prozesse mit der kürzeren Ausführungszeit zuerst an die Reihe kommen, müssen die anderen Prozesse nicht solange warten, bis der Prozess beendet ist und sie selbst dran kommen

12. Bei pre-emptive CPU Scheduling Strategien können Prozesse vorzeitig beendet (terminiert) werden.

Falsch - Prozesse können nur kurzeitig blockiert werden

13. Beim Priority Scheduling mit Aging wird "Aging" dazu verwendet, um das Alter des jeweiligen Prozesses anzugeben.

Falsch- Aging erhöht die Priorität 0=gut , hoch=schlecht

Priorität des laufenden Prozesses wird zwecks Fairness üblicherweise mit der Zeit verringert.

Betriebsysteme JA/NEIN Fragen

14. In interaktiven Betriebssystemen gibt es üblicherweise mehr context switches als in Stapelverarbeitungssystemen.
Richtig – bei Stapelverarbeitungssystemen wird versucht den Durchsatz (Jobs/h) zu maximieren um schneller mit den Aufgaben fertig zu werden. Bei interaktiven Betriebssystemen wird versucht die Abarbeitung von Prozessen möglichst „zeitgleich“ darzustellen, wodurch es öfters zu Context Switches kommt

15. Priority Scheduling minimiert die durchschnittliche Wartezeit pro Prozess.
Falsch – nur weil ein Prozess eine höhere Priorität hat, muss es nicht sein, dass er weniger Zyklen benötigt bis er fertig ist, wie andere Prozesse mit einer höheren Priorität